Using R projects

Here I want to talk briefly about the advantages of and tools for building project-oriented workflows. As a beginning R user, I was aware of the benefits of organizing each data analysis into a project: a folder on your computer that holds all the files relevant to that particular piece of work. However, I was unaware of tools such as Rprojects that allowed for easy navigation through projects.

What are R projects?

I stumbled across R projects when I was trying (and failing) to keep scripts in separate project directories organized AND accessible on multiple machines.

In my early days I was no stranger to having this at the top of each script:

```
setwd('path/that/only/ever/works/on/one/machine')
```

Here, R takes the absolute file path as an input then sets it as the current working directory of the R process. While `setwd()` served it's purpose, I often ended up frustrated when trying to change from project A to project B or working on my personal computer. And not to mention the impossibility of easily sharing code with this structure. Simply moving the entire directory to a new sub-folder or to a new drive breaks the links leaving you with a useless script!

RStudio fully supports Project-based workflows, making it easy to switch from one to another, have many projects open at once, re-launch recently used Projects, etc.

Creating a new project is simple and RStudio provides a walkthrough for this.
Here's a real-time video of how to create a new R Project yourself:

Now your working directory will always start at the same level your R project file exists.

```
getwd()
```

```
## [1] "/Users/joy/Documents/GitHub/code_club/slides/20210924_JN"
```

Whenever you refer to a file with a relative path, it will start by looking here.

Why use R projects

R projects are especially convenient if you want to simultaneously work in two project directories.
All you have to do is navigate to the folders associated with the projects, and open the `.Rproj` file.
Additionally, you can refer to files within this directory with relative path names.

```
readr::write_csv(diamonds, 'sample/diamonds.csv')
list.files('sample')
```

```
## [1] "diamonds.csv"
```

The here package

If you want to completely eliminate writing paths, I'd recommend checking out the `here` package.
`here` works hand in hand with Rprojects to allow you to easily build paths from the top-level directory.

```
library(here)
```

```
## here() starts at /Users/joy/Documents/GitHub/code_club/slides/20210924_JN
```

```
here::here()
```

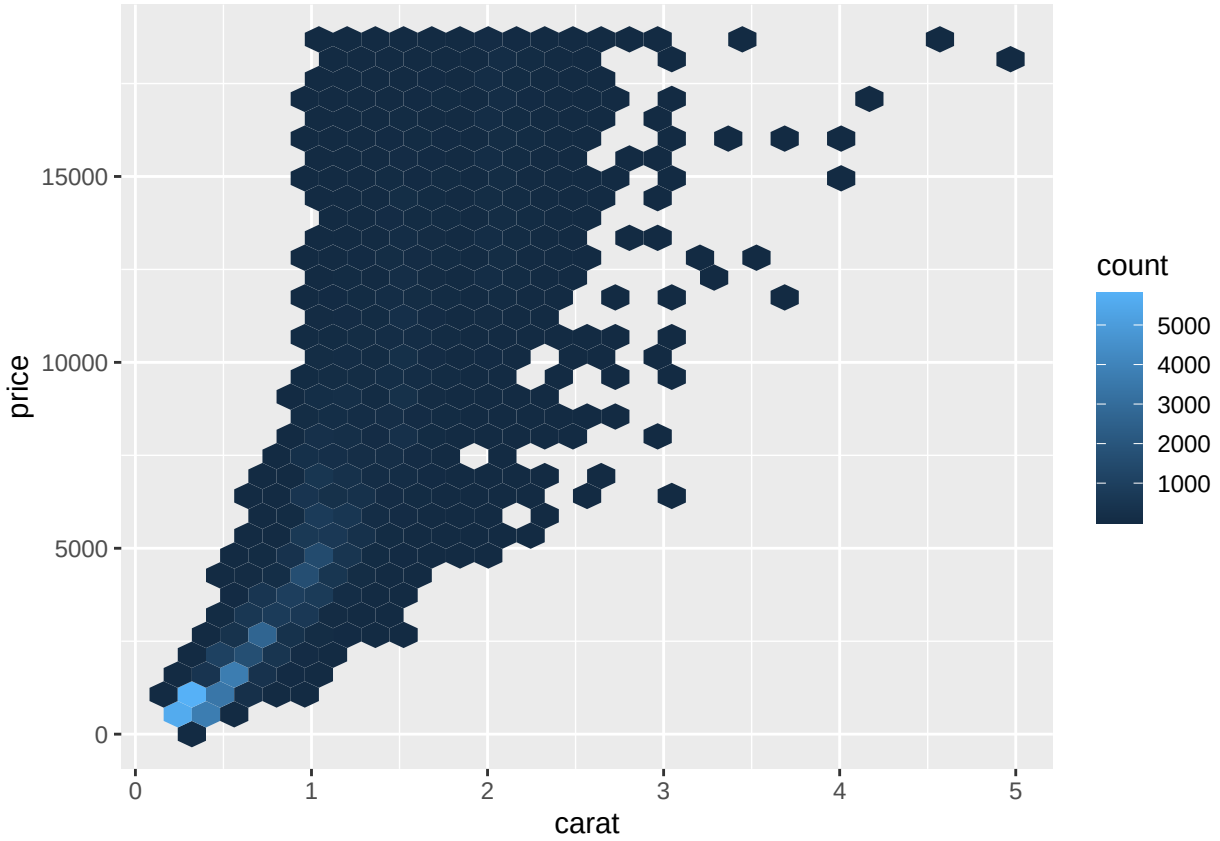
```
## [1] "/Users/joy/Documents/GitHub/code_club/slides/20210924_JN"
```

Now you can build paths relative to the top-level directory:

```
sample <- readr::read_csv(here::here('sample', 'diamonds.csv'))
```

```
## Parsed with column specification:
## cols(
##   carat = col_double(),
##   cut = col_character(),
##   color = col_character(),
##   clarity = col_character(),
##   depth = col_double(),
##   table = col_double(),
##   price = col_double(),
##   x = col_double(),
##   y = col_double(),
##   z = col_double()
## )
```

```
ggplot2::ggplot(sample, aes(carat, price)) +
  geom_hex()
```



Again – these relative paths will work regardless of where the associated source file exists within the project. Using this combination of `R projects` and `here` allows you to easily navigate through your project directory and subdirectories without ever writing another full path name!